Generalized Gradient Learning on Time Series under Elastic Transformations

Brijnesh J. Jain
Technische Universität Berlin, Germany
e-mail: brijnesh.jain@gmail.com

The majority of machine learning algorithms assumes that objects are represented as vectors. But often the objects we want to learn on are more naturally represented by other data structures such as sequences and time series. For these representations many standard learning algorithms are unavailable. We generalize gradient-based learning algorithms to time series under dynamic time warping. To this end, we introduce elastic functions, which extend functions on time series to matrix spaces. Necessary conditions are presented under which generalized gradient learning on time series is consistent. We indicate how results carry over to arbitrary elastic distance functions and to sequences consisting of symbolic elements. Specifically, four linear classifiers are extended to time series under dynamic time warping and applied to benchmark datasets. Results indicate that generalized gradient learning via elastic functions have the potential to complement the state-of-the-art in statistical pattern recognition on time series.

1. Introduction

Statistical pattern recognition on time series finds many applications in diverse domains such as speech recognition, medical signal analysis, and recognition of gestures [6, 7]. A challenge in learning on time series consists in filtering out the effects of shifts and distortions in time. A common and widely applied approach to address invariance of shifts and distortions are elastic transformations such as dynamic time warping (DTW). Following this approach amounts in learning on time series spaces equipped with an elastic proximity measure.

In comparison to Euclidean spaces, mathematical concepts such as the derivative of a function and a well-defined addition under elastic transformations are unknown in time series spaces. Therefore gradient-based algorithms can not be directly applied to time

series. The weak mathematical structure of time series spaces bears two consequences: (a) there are only few learning algorithms that directly operate on time series under elastic transformation; and (b) simple methods like the nearest neighbor classifier together with the DTW distance belong to the state-of-the-art and are reported to be *exceptionally difficult to beat* [1, 12, 29].

To advance the state-of-the-art in learning on time series, first adaptive methods have been proposed. They mainly devise or apply different measures of central tendency of a set of time series under dynamic time warping [11, 19, 20, 17]. The individual approaches reported in the literature are k-means [9, 14, 15, 18, 28], self-organizing maps [25], and learning vector quantization [25]. These methods have been formulated in a problem-solving manner without a unifying theme. Consequently, there is no link to a mathematical theory that allows us to (1) place existing adaptive methods in a proper context, (2) derive adaptive methods on time series other than those based on a concept of mean, and (3) prove convergence of adaptive methods to solutions that satisfy necessary conditions of optimality.

Here we propose generalized gradient methods on time series spaces that combine the advantages of gradient information and elastic transformation such that the above issues (1)–(3) are resolved. The key idea behind this approach is the concept of elastic function. Elastic functions extend functions on Euclidean spaces to time series spaces such that elastic transformations are preserved. Then learning on time series amounts in minimizing piecewise smooth risk functionals using generalized gradient methods proposed by [5, 16]. Specifically, we investigate elastic versions of logistic regression, (margin) perceptron learning, and linear support vector machine (SVM) for time series under dynamic time warping. We derive update rules and present different convergence results, in particular an elastic version of the perceptron convergence theorem. Though the main treatment focuses on univariate time series under DTW, we also show under which conditions the theory also holds for multivariate time series and sequences with non-numerical elements under arbitrary elastic transformations.

We tested the four elastic linear classifiers to all two-class problems of the UCR time series benchmark dataset [10]. The results show that elastic linear classifiers on time series behave similarly to linear classifiers on vectors. Furthermore, our findings indicate that generalized gradient learning on time series spaces have the potential to complement the state-of-the-art in statistical pattern recognition on time series, because the simplest elastic methods are already competitive with the best available methods.

The paper is organized as follows: Section 2 introduces background material. Section 3 proposes elastic functions, generalized gradient learning on sequence data, and elastic linear classifiers. In Section 4, we relate the proposed approach to previous approaches on averaging a set of time series. Section 5 presents and discusses experiments. Finally, Section 6 concludes with a summary of the main results and an outlook for further research.

2. Background

This section introduces basic material. Section 2.1 defines the DTW distance, Section 2.2 presents the problem of learning from examples, and Section 2.3 introduces piecewise smooth functions.

2.1. Dynamic Time Warping Distance

By $[n]$ we denote the set $\{1, \ldots, n\}$ for some $n \in \mathbb{N}$. A time series of length n is an ordered sequence $\boldsymbol{x} = (x_1, \ldots, x_n)$ with features $x_i \in \mathbb{R}$ sampled at discrete points of time $i \in [n]$.

To define the DTW distance between time series $\boldsymbol{x}$ and $\boldsymbol{y}$ of length n and m, resp., we construct a grid $\mathcal{G} = [n] \times [m]$. A warping path in grid $\mathcal{G}$ is a sequence $\phi = (\boldsymbol{t}_1, \ldots, \boldsymbol{t}_p)$ consisting of points $\boldsymbol{t}_k = (i_k, j_k) \in \mathcal{G}$ such that

1. $\boldsymbol{t}_1 = (1, 1)$ and $\boldsymbol{t}_p = (n, m)$ (boundary conditions)
2. $\boldsymbol{t}_{k+1} - \boldsymbol{t}_k \in \{(1, 0), (0, 1), (1, 1)\}$ (warping conditions)

for all $1 \leq k < p$.

A warping path ϕ defines an alignment between sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ by assigning elements x_i of sequence $\boldsymbol{x}$ to elements y_j of sequence $\boldsymbol{y}$ for every point $(i, j) \in \phi$. The boundary condition enforces that the first and last element of both time series are assigned to one another accordingly. The warping condition summarizes what is known as the monotonicity and continuity condition. The monotonicity condition demands that the points of a warping path are in strict ascending lexicographic order. The continuity condition defines the maximum step size between two successive points in a path.

The cost of aligning $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_m)$ along a warping path ϕ is defined by

$$d_\phi(\boldsymbol{x}, \boldsymbol{y}) = \sum_{(i,j) \in \phi} c\left(x_i, y_j\right),$$

where $c(x_i, y_j)$ is the local transformation cost of aligning features x_i and y_j. Unless otherwise stated, we assume that the local transformation costs are given by $c\left(x_i, y_j\right) = \left(x_i - y_j\right)^2$. Then the distance function

$$d(\boldsymbol{x}, \boldsymbol{y}) = \min_\phi \sqrt{d_\phi(\boldsymbol{x}, \boldsymbol{y})},$$

is the dynamic time warping (DTW) distance between $\boldsymbol{x}$ and $\boldsymbol{y}$, where the minimum is taken over all warping paths in $\mathcal{G}$.

2.2. The Problem of Learning

We consider learning from examples as the problem of minimizing a risk functional. To present the main ideas, it is sufficient to focus on supervised learning.

Consider an input space $\mathcal{X}$ and output space $\mathcal{Y}$. The problem of supervised learning is to estimate an unknown function $f_* : \mathcal{X} \to \mathcal{Y}$ on the basis of a training set

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \subseteq \mathcal{X} \times \mathcal{Y},$$

where the examples $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ are drawn independent and identically distributed according to a joint probability distribution $P(x, y)$ on $\mathcal{X} \times \mathcal{Y}$.

To measure how well a function $f : \mathcal{X} \to \mathcal{Y}$ predicts output values y from x, we introduce the risk

$$R[f] = \int_{\mathcal{X} \times \mathcal{Y}} \ell(y, f(x))\, dP(x, y),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a loss function that quantifies the cost of predicting $f(x)$ when the true output value is y.

The goal of learning is to find a function $f : \mathcal{X} \to \mathcal{Y}$ that minimizes the risk. The problem is that we can not directly compute the risk of f, because the probability distribution $P(x, y)$ is unknown. But we can use the training examples to estimate the risk of f by the empirical risk

$$R_N[f] = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, f(x_i)).$$

The empirical risk minimization principle suggests to approximate the unknown function f_* by a function

$$f_N = \arg\min_{f \in \mathcal{F}} R_N[f]$$

that minimizes the empirical risk over a fixed hypothesis space $\mathcal{F} \subset \mathcal{Y}^{\mathcal{X}}$ of functions $f : \mathcal{X} \to \mathcal{Y}$.

Under appropriate conditions on $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{F}$, the empirical risk minimization principle is justified in the following sense: (1) a minimizer f_N of the empirical risk exists, though it may not be unique; and (2) the risk $R[f_N]$ converges in probability to the risk $R[f_*]$ of the best but unknown function f_* when the number N of training examples goes to infinity.

2.3. Piecewise Smooth Functions

A function $f : \mathcal{X} \to \mathbb{R}$ defined on a Euclidean space $\mathcal{X}$ is piecewise smooth, if f is continuous and there is a finite collection of continuously differentiable functions $\mathcal{R}(f) = \{f_i : \mathcal{X} \to \mathbb{R} : i \in \mathcal{I}\}$ indexed by the set $\mathcal{I}$ such that

$$f(x) \in \{f_i(x) : i \in \mathcal{I}\}$$

for all $x \in \mathcal{X}$. We call the collection $\mathcal{R}(f)$ a representation for f. A function $f_i \in \mathcal{R}(f)$ satisfying $f_i(x) = f(x)$ is an active function of f at x. The set $\mathcal{A}(f, x) = \{i \in \mathcal{I} : f_i(x) = f(x)\}$ is the active index set of f at x. By

$$\partial f(x) = \{\nabla f_i(x) : i \in \mathcal{A}(f, x)\}$$

we denote the set of active gradients $\nabla f_i(x)$ of active function f_i at x. Active gradients are directional derivatives of f. At differentiable points x the set of active gradients is of the form $\partial f(x) = \{\nabla f(x)\}$.

Piecewise smooth functions are closed under composition, scalar multiplication, finite sums, pointwise max- and min-operations. In particular, the max- and min-operations of a finite collection of differentiable functions allow us to construct piecewise smooth functions. Piecewise functions f are non-differentiable on a set of Lebesgue measure zero, that is f is differentiable almost everywhere.

3. Generalized Gradient Learning on Time Series Spaces

This section generalizes gradient-based learning to time series spaces under elastic transformations. We first present the basic idea of the proposed approach in Section 3.1. Then Section 3.2 introduces the new concept of elastic functions. Based on this concept, Section 3.3 describes supervised generalized gradient learning on time series. As an example, Section 3.4 introduces elastic linear classifiers. In Section 3.5, we consider unsupervised generalized gradient learning. Section 3.6 sketches consistency results. Finally, Section 3.7 generalizes the proposed approach to other elastic proximity functions and arbitrary sequence data.

3.1. The Basic Idea

This section presents the basic idea of generalized gradient learning on time series. For this we assume that $\mathcal{F}_\mathcal{X}$ is a hypothesis space consisting of functions $F : \mathcal{X} \to \mathbb{R}$ defined on some Euclidean space $\mathcal{X}$. For example, $\mathcal{F}_\mathcal{X}$ consists of all linear functions on $\mathcal{X}$. First we show how to generalize functions $F \in \mathcal{F}_\mathcal{X}$ defined on Euclidean spaces to functions $f : \mathcal{T} \to \mathbb{R}$ on time series such that elastic transformations are preserved. The resulting functions f are called elastic. Then we turn the focus on learning an unknown elastic function over the new hypothesis space $\mathcal{F}_\mathcal{T}$ of elastic functions obtained from $\mathcal{F}_\mathcal{X}$.

We define elastic functions $f : \mathcal{T} \to \mathbb{R}$ on time series as a pullback of a function $F \in \mathcal{F}_\mathcal{X}$ by an embedding $\mu : \mathcal{T} \to \mathcal{X}$, that is $f(\boldsymbol{x}) = F(\mu(\boldsymbol{x}))$ for all time series $\boldsymbol{x} \in \mathcal{T}$.

In principle any injective map μ can be used. Here, we are interested in embeddings that preserve elastic transformations. For this, we select a problem-dependent base time series $\boldsymbol{z} \in \mathcal{T}$. Then we define an embedding $\mu_{\boldsymbol{z}} : \mathcal{T} \to \mathcal{X}$ that is isometric with respect to $\boldsymbol{z}$, that is

$$d(\boldsymbol{x}, \boldsymbol{z}) = \|\mu_{\boldsymbol{z}}(\boldsymbol{x}) - \mu_{\boldsymbol{z}}(\boldsymbol{z})\|$$

for all $\boldsymbol{x} \in \mathcal{T}$. It is important to note that an embedding $\mu_{\boldsymbol{z}}$ is distance preserving with respect to $\boldsymbol{z}$, only. In general, we will have $d(\boldsymbol{x}, \boldsymbol{y}) \leq \|\mu_{\boldsymbol{z}}(\boldsymbol{x}) - \mu_{\boldsymbol{z}}(\boldsymbol{y})\|$ showing that an embedding $\mu_{\boldsymbol{z}}$ will be an expansion of the time series space. This form of a restricted isometry turns out to be sufficient for our purposes. We call the pullback $f = F \circ \mu$ of F by μ elastic, if embedding μ preserves elastic distances with respect to some base time series. Figure 1 illustrates the concept of elastic function.

Next we show how to learn an unknown elastic function by risk minimization over the hypothesis space $\mathcal{F}_\mathcal{T}$ consisting of pullbacks of functions from $\mathcal{F}_\mathcal{X}$ by μ. For this we assume that $\Theta_\mathcal{T}$ is a set of parameters and the hypothesis space $\mathcal{F}_\mathcal{T}$ consists of functions $f_{\boldsymbol{\theta}}$ with parameter $\boldsymbol{\theta} \in \Theta_\mathcal{T}$. To convey the basic idea, we consider the simple case that the parameter set is of the form $\Theta_\mathcal{T} = \mathcal{T}$. Then the goal is to minimize a risk functional

$$\min_{\boldsymbol{\theta} \in \mathcal{T}} R[\boldsymbol{\theta}] \tag{1}$$

as a function of $\boldsymbol{\theta} \in \mathcal{T}$. We cast problem (1) to the equivalent problem

$$\min_{\boldsymbol{\theta} \in \mathcal{T}} R[\mu(\boldsymbol{\theta})], \tag{2}$$

Observe that the risk functional of problem (2) is a function of elements $\mu(\boldsymbol{\theta})$ from the Euclidean space $\mathcal{X}$. Since problem (2) is analytically difficult to handle, we consider the relaxed problem

$$\min_{\boldsymbol{\Theta} \in \mathcal{X}} R[\boldsymbol{\Theta}], \tag{3}$$

where the minimum is taken over the whole set $\mathcal{X}$, whereas problem (2) minimizes over the subset $\mu(\mathcal{T}) \subset \mathcal{X}$. The relaxed problem (3) is not only analytically more tractable but also learns a model from a larger hypothesis space and may therefore provide better asymptotical solutions, but may require more training data to reach acceptable test error rates [26].

3.2. Elastic Functions

This section formally introduces the concept of elastic function, which generalize functions on matrix spaces $\mathcal{X} = \mathbb{R}^{n \times m}$ to time series spaces. The matrix space $\mathcal{X}$ is the Euclidean space of all real $(n \times m)$-matrices with inner product

$$\langle \boldsymbol{X}, \boldsymbol{Y} \rangle = \sum_{i,j} x_{ij} \cdot y_{ij}.$$

for all $\boldsymbol{X}, \boldsymbol{Y} \in \mathcal{X}$. The inner product induces the Euclidean norm

$$\|\boldsymbol{X}\| = \sqrt{\langle \boldsymbol{X}, \boldsymbol{X} \rangle}$$

also known as the Frobenius norm.[1] The dimension $n \times m$ of $\mathcal{X}$ has the following meaning: the number n of rows refers to the maximum length of all time series from the training set $\mathcal{D}$. The number m of columns is a problem dependent parameter, called *elasticity* henceforth. A larger number m of columns admits higher elasticity and vice versa.

We first define an embedding from time series into the Euclidean space $\mathcal{X}$. We embed time series into a matrix from $\mathcal{X}$ along a warping path as illustrated in Figure

[1] We call $\|\boldsymbol{X}\|$ Euclidean norm to emphasize that we regard $\mathcal{X}$ as a Euclidean space.

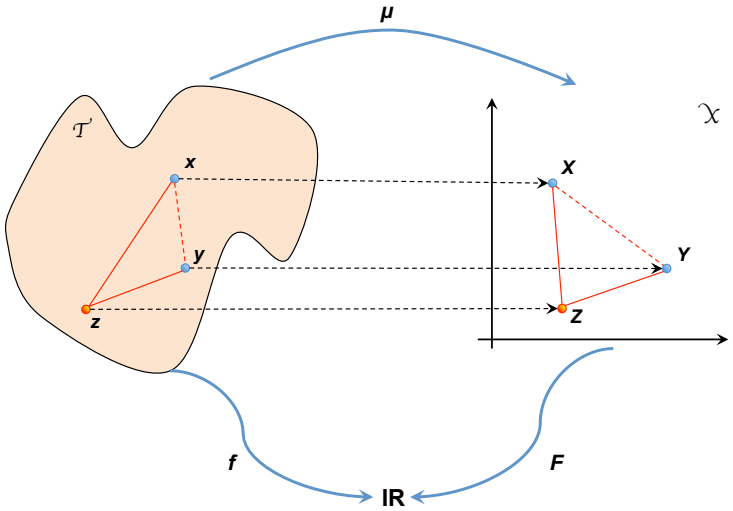

Figure 1: Illustration of elastic function $f : \mathcal{T} \to \mathbb{R}$ of a function $F : \mathcal{X} \to \mathbb{R}$. The map $\mu = \mu_{\boldsymbol{z}}$ embeds time series space $\mathcal{T}$ into the Euclidean space $\mathcal{X}$. Corresponding solid red lines indicate that distances between respective endpoints are preserved by μ. Corresponding dashed red lines show that distances between respective endpoints are not preserved. The diagram commutes, that is $f(\boldsymbol{x}) = F(\mu(\boldsymbol{x}))$ is a pullback of F by μ.

2. Suppose that $\boldsymbol{x} = (x_1, \ldots, x_k)$ is a time series of length $k \leq n$. By $\mathcal{P}(\boldsymbol{x})$ we denote the set of all warping paths in the grid $\mathcal{G} = [k] \times [m]$ defined by the length k of $\boldsymbol{x}$ and elasticity m. An elastic embedding of time series $\boldsymbol{x}$ into matrix $\boldsymbol{Z} = (z_{ij})$ along warping path $\phi \in \mathcal{P}(\boldsymbol{x})$ is a matrix $\boldsymbol{x} \otimes_\phi \boldsymbol{Z} = (x_{ij})$ with elements

$$x_{ij} = \begin{cases} x_i & : \quad (i,j) \in \phi \\ z_{ij} & : \quad \text{otherwise} \end{cases} .$$

Suppose that $F : \mathcal{X} \to \mathbb{R}$ is a function defined on the Euclidean space $\mathcal{X}$. An elastic function of F based on matrix $\boldsymbol{Z}$ is a function $f : \mathcal{T} \to \mathbb{R}$ with the following property: for every time series $\boldsymbol{x} \in \mathcal{T}$ there is a warping path $\phi \in \mathcal{P}(\boldsymbol{x})$ such that

$$f(\boldsymbol{x}) = F(\boldsymbol{x} \otimes_\phi \boldsymbol{Z}).$$

The representation set and active set of f at $\boldsymbol{x}$ are of the form

$$\begin{aligned} \mathcal{R}(f, \boldsymbol{x}) &= \{F(\boldsymbol{x} \otimes_\phi \boldsymbol{Z}) \ : \ \phi \in \mathcal{P}(\boldsymbol{x})\} \\ \mathcal{A}(f, \boldsymbol{x}) &= \{\phi \in \mathcal{P}(\boldsymbol{x}) \ : \ f(\boldsymbol{x}) = F(\boldsymbol{x} \otimes_\phi \boldsymbol{Z})\} . \end{aligned}$$

The definition of elastic function corresponds to the properties described in Section 3.1 and in Figure 1. To see this, we define an embedding $\mu_{\boldsymbol{Z}} : \mathcal{T} \to \mathcal{X}$ that first selects

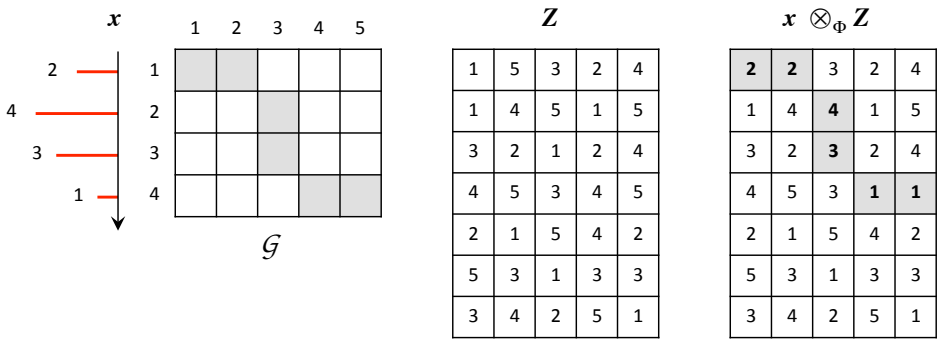

Figure 2: Embedding of time series $\boldsymbol{x} = (2, 4, 3, 1)$ into matrix $\boldsymbol{Z}$ along warping path ϕ. From left to right: Time series $\boldsymbol{x}$, grid $\mathcal{G}$ with highlighted warping path ϕ, matrix $\boldsymbol{Z}$, and matrix $\boldsymbol{x} \otimes_\phi \boldsymbol{Z}$ obtained after embedding $\boldsymbol{x}$ into $\boldsymbol{Z}$ along ϕ. We assume that the length of the longest time series in the training set is $n = 7$. Therefore the matrix $\boldsymbol{Z}$ has $n = 7$ rows. The number m of columns of $\boldsymbol{Z}$ is a problem dependent parameter and set to $m = 5$ in this example. Since time series $\boldsymbol{x}$ has length $k = 4$, the grid $\mathcal{G} = [k] \times [m]$ containing all feasible warping paths consists of 4 rows and 5 columns. Grids $\mathcal{G}$ vary only in the number k of rows in accordance with the length $k \leq n$ of the time series to be embedded, but always have m columns.

for every time series $\boldsymbol{x}$ an active warping path $\phi \in \mathcal{A}(f, \boldsymbol{x})$ and then maps $\boldsymbol{x}$ to the matrix $\mu_{\boldsymbol{Z}}(\boldsymbol{x}) = \boldsymbol{x} \otimes_\phi \boldsymbol{Z}$. Then we have $F(\mu_{\boldsymbol{Z}}(\boldsymbol{x})) = f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{T}$. Suppose that the rows of matrix $\boldsymbol{Z}$ are all equal to $\boldsymbol{z}$. Then $\mu_{\boldsymbol{z}} = \mu_{\boldsymbol{Z}}$ is isometric with respect to $\boldsymbol{z}$.

Next, we consider examples of elastic functions. The first two examples are fundamental for extending a broad class of gradient-based learning algorithms to time series spaces.

Example 1 (Elastic Euclidean Distance) *Let $\boldsymbol{Y} \in \mathcal{X}$. Consider the function*

$$D_{\boldsymbol{Y}} : \mathcal{X} \to \mathbb{R}_+, \quad \boldsymbol{X} \mapsto \|\boldsymbol{X} - \boldsymbol{Y}\|$$

Then

$$\delta_{\boldsymbol{Y}} : \mathcal{T} \to \mathbb{R}_+, \quad \boldsymbol{x} \mapsto \min_{\phi \in \mathcal{P}(\boldsymbol{x})} \|\boldsymbol{x} \otimes_\phi \boldsymbol{Y} - \boldsymbol{Y}\|,$$

is an elastic function of $D_{\boldsymbol{Y}}$. To see this, observe that from

$$\delta_{\boldsymbol{Y}}(\boldsymbol{x}) = \min_{\phi \in \mathcal{P}(\boldsymbol{x})} \|\boldsymbol{x} \otimes_\phi \boldsymbol{Y} - \boldsymbol{Y}\| = \min_{\phi \in \mathcal{P}(\boldsymbol{x})} D_{\boldsymbol{Y}}(\boldsymbol{x} \otimes_\phi \boldsymbol{Y})$$

follows $\delta_{\boldsymbol{Y}}(\boldsymbol{x}) \in \mathcal{R}(\delta_{\boldsymbol{Y}}, \boldsymbol{x}) = \{D_{\boldsymbol{Y}}(\boldsymbol{x} \otimes_\phi \boldsymbol{Y}) : \phi \in \mathcal{P}(\boldsymbol{x})\}$. See Figure 3 for an illustration. We call $\delta_{\boldsymbol{Y}}$ elastic Euclidean distance *with parameter $\boldsymbol{Y}$.* ■

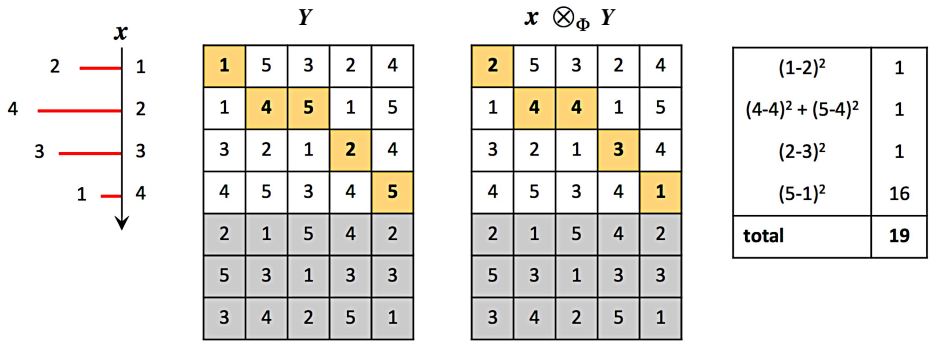

Figure 3: Elastic Euclidean distance $\delta_{\boldsymbol{Y}}(\boldsymbol{x})$. From left to right: time series $\boldsymbol{x} = (2, 4, 3, 1)$, matrix $\boldsymbol{Y}$, matrix $\boldsymbol{x} \otimes_\phi \boldsymbol{Y}$ obtained by embedding $\boldsymbol{x}$ into matrix $\boldsymbol{Y}$ along optimal warping path ϕ, and distance computation by aggregating the local costs giving $\delta_{\boldsymbol{Y}}(\boldsymbol{x}) = \sqrt{19}$. The optimal path is highlighted in orange in $\boldsymbol{Y}$ and in $\boldsymbol{x} \otimes_\phi \boldsymbol{Y}$. Gray shaded areas in both matrices refer to parts that are not used, because the length $k = 4$ of $\boldsymbol{x}$ is less than $n = 7$. Since $\boldsymbol{x}$ is embedded into $\boldsymbol{Y}$ only elements lying on the path ϕ contribute to the distance. All other local cost between elements of $\boldsymbol{Y}$ and $\boldsymbol{x} \otimes_\phi \boldsymbol{Y}$ are zero.

Example 2 (Elastic Inner Product) *Let $\boldsymbol{W} \in \mathcal{X}$. Consider the function*

$$S_{\boldsymbol{W}} : \mathcal{X} \to \mathbb{R}, \quad \boldsymbol{X} \mapsto \langle \boldsymbol{X}, \boldsymbol{W} \rangle$$

Then the function

$$\sigma_{\boldsymbol{W}} : \mathcal{T} \to \mathbb{R}, \quad \boldsymbol{x} \mapsto \max_{\phi \in \mathcal{P}(\boldsymbol{x})} \langle \boldsymbol{x} \otimes_\phi \boldsymbol{0}, \boldsymbol{W} \rangle,$$

is an elastic function of $S_{\boldsymbol{W}}$, called elastic inner product *with parameter $\boldsymbol{W}$.* ■

The elastic Euclidean distance and elastic inner product are elastic proximities closely related to the DTW distance, where the elastic Euclidean distance generalizes the DTW distance. The time and space complexity of both elastic proximities are $O(nm)$. If no optimal warping path is required, space complexity can be reduced to $O(\max(n, m))$. To see this, we refer to Algorithm 1. To obtain an optimal warping path, we can trace-back along the score matrix $\boldsymbol{S}$ in the usual way. The procedure in Algorithm 1 applies exactly the same dynamic programming scheme as the one for the standard DTW distance and therefore has the same time and space complexity.

Observe that both elastic proximities embed time series into different matrices. Elastic Euclidean distances embed time series into the parameter matrix and elastic inner products always embed time series into the zero-matrix $\boldsymbol{0}$.

Algorithm 1 (Elastic Inner Product)

Input:

– time series $\boldsymbol{x} = (x_1, \ldots, x_k)$ with $k \leq n$

– elasticity m

– weight matrix $\boldsymbol{W} = (w_{ij}) \in \mathbb{R}^{n \times m}$

Procedure:

Let $\boldsymbol{S} = (s_{ij}) \in \mathbb{R}^{k \times m}$ be the initial score matrix

$s_{11} \leftarrow x_1 w_{11}$

for $i = 2$ **to** k **do**

$s_{i1} \leftarrow s_{i-1,1} + x_i w_{i1}$

for $j = 2$ **to** m **do**

$s_{1j} \leftarrow s_{1,j-1} + x_1 w_{1j}$

for $i = 2$ **to** k **do**

for $j = 2$ **to** m **do**

$s_{ij} = x_i w_{ij} + \max\{s_{i-1,j}, s_{i,j-1}, s_{i-1,j-1}\}$

Return:

– $\sigma_{\boldsymbol{W}}(\boldsymbol{x}) = s_{km}$

Remark: This algorithm can also be used to compute elastic Euclidean distances. For this, replace all products $x_i w_{ij}$ by squared costs $(x_i - w_{ij})^2$ and the max-operation by a min-operation.

Example 3 (Elastic Linear Function) *Let $\Theta = \mathcal{X} \times \mathbb{R}$ be a set of parameters and let $\boldsymbol{\theta} = (\boldsymbol{W}, b) \in \Theta$ be a parameter. Consider the linear function*

$$F_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}, \quad \boldsymbol{X} \;\mapsto\; b + S_{\boldsymbol{W}}(\boldsymbol{X}) = b + \langle \boldsymbol{X}, \boldsymbol{W} \rangle,$$

where $\boldsymbol{W}$ is the weight matrix and b is the bias. The function

$$f_{\boldsymbol{\theta}} : \mathcal{T} \to \mathbb{R}, \quad \boldsymbol{x} \;\mapsto\; b + \sigma_{\boldsymbol{W}}(\boldsymbol{x}),$$

is an elastic function of $F_{\boldsymbol{\theta}}$, called elastic linear function*.* ■

Example 4 (Single-Layer Neural Network) *Let $\Theta = \mathcal{X}^r \times \mathbb{R}^{2r+1}$ be a set of parameters. Consider the function*

$$f_{\boldsymbol{\theta}} : \mathcal{T} \to \mathbb{R}, \quad \boldsymbol{x} \;\mapsto\; b + \sum_{i=1}^{r} w_i \, \alpha(f_i(\boldsymbol{x})),$$

where $\alpha(z)$ is a sigmoid function, $f_i = f_{\boldsymbol{\theta}_i}$ are elastic linear functions with parameters $\boldsymbol{\theta}_i = (\boldsymbol{W}_i, b_i)$, and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_r, w_1, \ldots, w_r, b)$. The function $f_{\boldsymbol{\theta}}$ implements an elastic neural network for time series with r sigmoid units in the hidden layer and a single linear unit in the output layer. ■

3.3. Supervised Generalized Gradient Learning

This section introduces a generic scheme of generalized gradient learning for time series under dynamic time warping.

Let $\Theta = \mathcal{X}^r \times \mathbb{R}^s$ be a set of parameters. Consider a hypothesis space $\mathcal{F}$ of functions $f_{\boldsymbol{\theta}} : \mathcal{T} \to \mathcal{Y}$ with parameter $\boldsymbol{\theta} = (\boldsymbol{W}_1, \ldots, \boldsymbol{W}_r, \boldsymbol{b}) \in \Theta$. Suppose that $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N), y_N\} \subseteq \mathcal{T} \times \mathcal{Y}$ is a training set. According to the empirical risk minimization principle, the goal is to minimize

$$R_N[\boldsymbol{\theta}] = R_N[f_{\boldsymbol{\theta}}] = \sum_{i=1}^{N} \ell(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))$$

as a function of $\boldsymbol{\theta}$. Since R_N is a function of $\boldsymbol{\theta}$, we rewrite the loss by interchanging the role of argument $\boldsymbol{z} = (\boldsymbol{x}, y)$ and parameter $\boldsymbol{\theta}$ such that

$$\ell_{\boldsymbol{z}} : \Theta \to \mathbb{R}, \quad \boldsymbol{\theta} \mapsto \ell(y, f_{\boldsymbol{\theta}}(\boldsymbol{x})). \tag{4}$$

We assume that the loss $\ell_{\boldsymbol{z}}$ is piecewise smooth with representation set

$$\mathcal{R}(\ell_{\boldsymbol{z}}) = \{\ell_{\boldsymbol{\Phi}} : \Theta \to \mathbb{R} : \boldsymbol{\Phi} = (\phi_1, \ldots, \phi_r) \in \mathcal{P}^r(\boldsymbol{x})\}$$

indexed by r-tuples of warping paths from $\mathcal{P}(\boldsymbol{x})$. The gradient $\nabla \ell_{\boldsymbol{\Phi}}$ of an active function $\ell_{\boldsymbol{\Phi}}$ at $\boldsymbol{\theta}$ is given by

$$\nabla \ell_{\boldsymbol{\Phi}} = \left(\frac{\partial \ell_{\boldsymbol{\Phi}}}{\partial \boldsymbol{W}_1}, \ldots, \frac{\partial \ell_{\boldsymbol{\Phi}}}{\partial \boldsymbol{W}_r}, \frac{\partial \ell_{\boldsymbol{\Phi}}}{\partial \boldsymbol{b}} \right),$$

where $\partial \ell_{\boldsymbol{\Phi}} / \partial \boldsymbol{\theta}_i$ denotes the partial derivative of $\ell_{\boldsymbol{\Phi}}$ with respect to $\boldsymbol{\theta}_i$. The incremental update rule of the generalized gradient method is of the form

$$\boldsymbol{W}_i^{t+1} = \boldsymbol{W}_i^t - \eta^t \cdot \frac{\partial}{\partial \boldsymbol{W}_i^t} \ell_{\boldsymbol{\Phi}}(\boldsymbol{\theta}^t) \tag{5}$$

$$\boldsymbol{b}^{t+1} = \boldsymbol{b}^t - \eta^t \cdot \frac{\partial}{\partial \boldsymbol{b}^t} \ell_{\boldsymbol{\Phi}}(\boldsymbol{\theta}^t) \tag{6}$$

for all $i \in [r]$. Section 3.6 discusses consistency of variants of update rule (5) and (6).

3.4. Elastic Linear Classifiers

Let $\mathcal{Y} = \{\pm 1\}$ be the output space consisting of two class labels. An elastic linear classifier is a function of the form

$$h_{\boldsymbol{\theta}} : \mathcal{T} \to \mathcal{Y}, \quad \boldsymbol{x} \mapsto \begin{cases} +1 & : \quad f_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0 \\ -1 & : \quad f_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0 \end{cases} \tag{7}$$

where $f_{\boldsymbol{\theta}}(\boldsymbol{x}) = b + \sigma_{\boldsymbol{W}}(\boldsymbol{x})$ is an elastic linear function and $\boldsymbol{\theta} = (\boldsymbol{W}, b)$ summarizes the parameters. We assign a time series $\boldsymbol{x}$ to the positive class if $f_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0$ and to the negative class otherwise.

Elastic Logistic Regression		$\mathcal{Y} = \{0, 1\}$
logistic function	$g_{\boldsymbol{\theta}}(\boldsymbol{x}) = 1/\left(1 + \exp(-f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)$	
loss function	$\ell = -y \log(g_{\boldsymbol{\theta}}(\boldsymbol{x})) - (1-y)\log(1 - g_{\boldsymbol{\theta}}(\boldsymbol{x}))$	
partial derivative	$\partial_{\boldsymbol{W}} \ell = -\left(y - g_{\boldsymbol{\theta}}(\boldsymbol{x})\right)) \cdot \boldsymbol{X}$	
Elastic Perceptron		$\mathcal{Y} = \{\pm 1\}$
loss function	$\ell = \max\{0, -y \cdot f_{\boldsymbol{\theta}}(\boldsymbol{x})\}$	
partial derivative	$\partial_{\boldsymbol{W}} \ell = -y \cdot \boldsymbol{X} \cdot \mathbb{I}_{\{\ell > 0\}}$	
Elastic Margin Perceptron		$\mathcal{Y} = \{\pm 1\}$
loss function	$\ell = \max\{0, \xi - y \cdot f_{\boldsymbol{\theta}}(\boldsymbol{x})\}$	
partial derivative	$\partial_{\boldsymbol{W}} \ell = -y \cdot \boldsymbol{X} \cdot \mathbb{I}_{\{\ell > 0\}}$	
Elastic Linear SVM		$\mathcal{Y} = \{\pm 1\}$
loss function	$\ell = \lambda \left\|\boldsymbol{W}\right\|^2 + \max\{0, 1 - y \cdot f_{\boldsymbol{\theta}}(\boldsymbol{x})\}$	
partial derivative	$\partial_{\boldsymbol{W}} \ell = -y \cdot \boldsymbol{X} \cdot \mathbb{I}_{\{\ell > 0\}}$	

Table 1: Examples of elastic linear classifiers. By $\partial_{\boldsymbol{W}} \ell$ we denote a partial derivative of an active function of ℓ with respect to $\boldsymbol{W}$. The partial derivatives $\partial_b \ell$ coincide with their corresponding counterparts in vector spaces and are therefore not included. The matrix $\boldsymbol{X} = \boldsymbol{x} \otimes_\phi \boldsymbol{0}$ is obtained by embedding time series $\boldsymbol{x}$ into the zero-matrix $\boldsymbol{0}$ along active warping path ϕ. The indicator function $\mathbb{I}_{\{z\}}$ returns 1 if the boolean expression z is true and returns 0, otherwise. The elastic perceptron is a special case of elastic margin perceptron with margin $\xi = 0$. The elastic linear SVM can be regarded as a special L_2-regularized elastic margin perceptron with margin $\xi = 1$.

Depending on the choice of loss function $\ell(y, f_{\boldsymbol{\theta}}(\boldsymbol{x}))$, we obtain different elastic linear classifiers as shown in Table 1. The loss function of elastic logistic regression is differentiable as a function of $f_{\boldsymbol{\theta}}$ and b, but piecewise smooth as a function of $\boldsymbol{W}$. All other loss functions are piecewise smooth as a function of $f_{\boldsymbol{\theta}}$, b and $\boldsymbol{W}$.

From the partial derivatives, we can construct the update rule of the generalized gradient method. For example, the incremental / stochastic update rule of the elastic perceptron is of the form

$$\boldsymbol{W}^{t+1} = \boldsymbol{W}^t + \eta^t \, y \boldsymbol{X} \tag{8}$$

$$b^{t+1} = b^t + \eta^t \, y, \tag{9}$$

where $(\boldsymbol{x}, y)$ is the training example at iteration t, and $\boldsymbol{X} = \boldsymbol{x} \otimes_\phi \boldsymbol{0}$ with $\phi \in \mathcal{A}(\ell, \boldsymbol{x})$. From the factor $\mathbb{I}_{\{\ell > 0\}}$ shown in Table 1 follows that the update rule given in (8) and (9) is only applied when $\boldsymbol{x}$ is misclassified.

We present three convergence results. A proof is given in Appendix A.

Convergence of the generalized gradient method. The generalized gradient method for

minimizing the empirical risk of an elastic linear classifier with convex loss converges to a local minimum under the assumptions of [5], Theorem 4.1.

Convergence of the stochastic generalized gradient method. This method converges to a local minimum of the expected risk of an elastic linear classifier with convex loss under the assumptions of [5], Theorem 5.1.

Elastic margin perceptron convergence theorem. The perceptron convergence theorem states that the perceptron algorithm with constant learning rate finds a separating hyperplane, whenever the training patterns are linearly separable. A similar result holds for the elastic margin perceptron algorithm.

A finite training set $\mathcal{D} \subseteq \mathcal{T} \times \mathcal{Y}$ is elastic-linearly separable, if there are parameters $\boldsymbol{\theta} = (\boldsymbol{W}, b)$ such that $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = y$ for all examples $(\boldsymbol{x}, y) \in \mathcal{D}$. We say, $\mathcal{D}$ is elastic-linearly separable with margin $\xi > 0$ if

$$\min_{(\boldsymbol{x},y)\in\mathcal{D}} \; y\,(b + \sigma(\boldsymbol{x}, \boldsymbol{W})) \geq \xi.$$

Then the following convergence theorem holds:

Theorem 1 (Elastic Margin Perceptron Convergence Theorem) *Suppose that $\mathcal{D} \subseteq \mathcal{T} \times \mathcal{Y}$ is elastic-linearly separable with margin $\xi > 0$. Then the elastic margin perceptron algorithm with fixed learning rate η and margin-parameter $\lambda \leq \xi$ converges to a solution $(\boldsymbol{W}, b)$ that correctly classifies the training examples from $\mathcal{D}$ after a finite number of update steps, provided the learning rate is chosen sufficiently small.*

3.5. Unsupervised Generalized Gradient Learning

Several unsupervised learning algorithms such as, for example, k-means, self-organizing maps, principal component analysis, and mixture of Gaussians are based on the concept of (weighted) mean. Once we know how to average a set of time series, extension of mean-based learning methods to time series follows the same rules as for vectors. Therefore, it is sufficient to focus on the problem of averaging a set of time series.

Suppose that $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathcal{T}$ is a set of unlabeled time series. Consider the sum of squared distances

$$F(\boldsymbol{Y}) = \sum_{i=1}^{N} \min \left\{ \|\boldsymbol{x}_i \otimes_{\phi_i} \boldsymbol{Y} - \boldsymbol{Y}\|^2 \,:\, \phi_i \in \mathcal{P}(\boldsymbol{x}_i) \right\}. \tag{10}$$

A matrix $\boldsymbol{Y}_*$ that minimizes F is a mean of the set $\mathcal{D}$ and the minimum value $F_* = F(\boldsymbol{Y}_*)$ is the variation of $\mathcal{D}$. The update rule of the generalized gradient method is of the form

$$\boldsymbol{Y}^{t+1} = \boldsymbol{Y}^t - \eta^t \sum_{i=1}^{N} \left(\boldsymbol{X}_i - \boldsymbol{Y}^t\right), \tag{11}$$

where $\boldsymbol{X}_i = \boldsymbol{x}_i \otimes_{\phi_i} \boldsymbol{Y}^t$ is the matrix obtained by embedding the i-th training example $\boldsymbol{x}_i$ into matrix $\boldsymbol{Y}^t$ along active warping path ϕ_i. Under the conditions of [5], Theorem 4.1, the generalized gradient method for minimizing f using update rule (11) is consistent in the mean and variation.

We consider the special case, when the learning rate is constant and takes the form $\eta^t = 1/N$ for all $t \geq 0$. Then update rule (11) is equivalent to

$$\boldsymbol{Y}^{t+1} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{X}_i, \tag{12}$$

where $\boldsymbol{X}_i$ is as in (11).

3.6. A Note on Convergence and Consistency

Gradient-based methods in statistical pattern recognition typically assume that the functions of the underlying hypothesis space is differentiable. However, many loss functions in machine learning are piecewise smooth, such as, for example, the loss of perceptron learning, k-means, and loss functions using ℓ_1-regularization. This case has been discussed and analyzed by [2].

When learning in elastic spaces, hypothesis spaces consist of piecewise smooth functions, which are pullbacks of smooth functions. Since piecewise smooth functions are closed under composition, the situation is similar as in standard pattern recognition, where hypothesis spaces consist of smooth functions. What has changed is that we will have "more" non-smooth points. Nevertheless, the set of non-smooth points remains negligible in the sense that it forms a set of Lebesgue measure zero.

Piecewise smooth functions are locally Lipschitz and therefore admit a Clarke's subdifferential Df at each point [3]. A Clarke's subdifferential Df is a set that contains elements, called generalized gradients. At differentiable points, the Clarke subdifferential coincides with the gradient, that is $Df(x) = \{\nabla f(x)\}$. A necessary condition of optimality of f at x is $0 \in Df(x)$.

Using these and other concepts from non-smooth analysis, we can construct minimization procedures that generalize gradient descent methods. In previous subsections, we presented a slightly simpler variant of the following generalized gradient method: Consider the minimization problem

$$\min_{x \in \mathcal{Z}} \quad f(x), \tag{13}$$

where f is a piecewise smooth function and $\mathcal{Z} \subseteq \mathcal{X}$ is a bounded convex constraint set. Let $\mathcal{Z}_*$ denote the subset of solutions satisfying the necessary condition of optimality and $f(\mathcal{Z}_*) = \{f(x) : x \in \mathcal{Z}_*\}$ is the set of solution values. Consider the following iterative method:

$$x^0 \in \mathcal{Z} \tag{14}$$

$$x^{t+1} \in \Pi_{\mathcal{Z}} \left(x^t - \eta^t \cdot g^t\right), \tag{15}$$

where $g^t \in Df(x^t)$ is a generalized gradient of f at x^t, $\Pi_{\mathcal{Z}}$ is the multi-valued projection onto $\mathcal{Z}$ and η^t is the learning rate satisfying the conditions

$$\lim_{t\to\infty} \eta^t = 0 \qquad \text{and} \qquad \sum_{t=0}^{\infty} \eta^t = \infty. \tag{16}$$

The generalized gradient method (14)–(16) minimizes a piecewise smooth function f by selecting a generalized gradient, performing the usual update step, and then projects the updated point to the constraint set $\mathcal{Z}$. If f is differentiable at x^t, which is almost always the case, then the update amounts to selecting an active index $i \in \mathcal{A}(f, x)$ of f at the current iterate x^t and then performing gradient descent along direction $-\nabla f_i(x^t)$.

Note that the constraint set $\mathcal{Z}$ has been ignored in previous subsections. We introduce a sufficiently large constraint set $\mathcal{Z}$ to ensure convergence. In a practical setting, we may ignore specifying $\mathcal{Z}$ unless the sequence (x^t) accidentally goes to infinity.

Under mild additional assumptions, this procedure converges to a solution satisfying the necessary condition of optimality [5], Theorem 4.1: *The sequence* (x^t) *generated by method* (14)–(16) *converges to the solution of problem* (13) *in the following sense:*

1. *the limits points* $\bar{x}$ *of* (x^t) *with minimum value* $f(\bar{x})$ *are contained in* $\mathcal{Z}_*$.
2. *the limits points* $\bar{f}$ *of* $(f(x^t))$ *are contained in* $f(\mathcal{Z}_*)$.

Consistency of the stochastic generalized gradient method for minimizing the expected risk functional follows from [5], Theorem 5.1, provided similar assumptions are satisfied.

3.7. Generalizations

This section indicates some generalizations of the concept of elastic functions.

3.7.1. Generalization to other Elastic Distance Functions

Elastic functions as introduced here are based on the DTW distance via embeddings along a set of feasible warping paths with squared differences as local transformation costs. The choice of distance function and local transformation cost is arbitrary. We can equally well define elastic functions based on proximities other than the DTW distance. Results on learning carry over whenever a proximity ρ on time series satisfies the following sufficient conditions: (1) ρ minimizes the costs over a set of feasible paths, (2) the cost of a feasible path is a piecewise smooth function as a function of the local transformation costs, and (3) the local transformation costs are piecewise smooth.

With regard to the DTW distance, these generalizations include the Euclidean distance and DTW distances with additional constraints such as the Sakoe-Chiba band [23]. Furthermore, absolute differences as local transformation cost are feasible, because the absolute value function is piecewise smooth.

3.7.2. Generalization to Multivariate Time Series

A multivariate time series is an ordered sequence $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ consisting of feature vectors $\boldsymbol{x}_i \in \mathbb{R}^d$. We can define the DTW distance between multivariate time series $\boldsymbol{x}$ and $\boldsymbol{y}$ as in the univariate case but replace the local transformation cost $c(x_i, y_j) = (x_i - y_j)^2$ by $c(\boldsymbol{x}_i, \boldsymbol{y}_j) = \|\boldsymbol{x}_i - \boldsymbol{y}_j\|^2$.

To define elastic functions, we embed multivariate time series into the set $\mathcal{X} = (\mathbb{R}^d)^{n \times m}$ of vector-valued matrices $\boldsymbol{X} = (\boldsymbol{x}_{ij})$ with elements $\boldsymbol{x}_{ij} \in \mathbb{R}^d$. These adjustment preserve piecewise smoothness, because the Euclidean space $\mathcal{X}$ is a direct product of lower-dimensional Euclidean spaces.

3.7.3. Generalization to Sequences with Symbolic Attributes

We consider sequences $\boldsymbol{x} = (x_1, \ldots, x_n)$ with attributes x_i from some finite set $\mathcal{A}$ of d attributes (symbols). Since $\mathcal{A}$ is finite, we can represent its attributes $a \in \mathcal{A}$ by d-dimensional binary vectors $\boldsymbol{a} \in \{0, 1\}^d$, where all but one element is zero. The unique non-zero element has value one and is related to attribute a. In doing so, we can reduce the case of attributed sequences to the case of multivariate time series.

We can introduce the following local transformation costs

$$c(x_i, y_j) = \begin{cases} 0 & : \quad x_i = y_j \\ 1 & : \quad x_i \neq y_j \end{cases} .$$

More generally, we can define local transformation costs of the form

$$c(x_i, y_j) = k(x_i, x_i) - 2k(x_i, y_j) + k(y_j, y_j),$$

where $k : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ is a positive-definite kernel. Provided that the kernel is an inner product in some finite-dimensional feature space, we can reduce this generalization also to the case of multivariate time series.

4. Relationship to Previous Approaches

Previous work on adaptive methods either focus on computing or are based on a concept of (weighted) mean of a set of time series. Most of the literature is summarized in [11, 17, 18, 25]. To place those approaches into the framework of elastic functions, it is sufficient to consider the problem of computing a mean of a set of time series.

Suppose that $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is a set of time series. A mean is any time series $\boldsymbol{y}_*$ that minimizes the sum of squared DTW distances

$$f(\boldsymbol{y}) = \sum_{i=1}^{N} d^2(\boldsymbol{x}_i, \boldsymbol{y}).$$

Algorithm 2 outlines a unifying minimization procedure of f. The set $\mathcal{Z}$ in line 1 of the procedure consists of all matrices with n identical rows, where n is the maximum length of all time series from $\mathcal{D}$. Thus, there is a one-to-one correspondence between

Algorithm 2 (Mean Computation)

Input:

– sample $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \mathcal{T}$

Procedure:

1. initialize $\boldsymbol{Y} \in \mathcal{Z}$
2. **repeat**
   2.1. determine active warping paths ϕ_i that embed $\boldsymbol{x}_i$ into $\boldsymbol{Y}$
   2.2. update $\boldsymbol{Y} \leftarrow \upsilon(\boldsymbol{Y}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \phi_1, \ldots, \phi_N)$
   2.3. project $\boldsymbol{Y} \leftarrow \pi(\boldsymbol{Y})$ to $\mathcal{Z}$

   until convergence

Return:

– approximation $\boldsymbol{y}$ of mean

time series from $\mathcal{T}$ and matrices from the subset $\mathcal{Z}$. By construction, we have $f(\boldsymbol{y}) = F(\boldsymbol{Y})$, where $\boldsymbol{Y} \in \mathcal{Z}$ is the matrix with all rows equal to $\boldsymbol{y}$ and $F(\boldsymbol{Y})$ is as defined in eq. (10).

In line 2.1, we determine active warping paths of the function $F(\boldsymbol{Y})$ that embed $\boldsymbol{x}_i$ into matrix $\boldsymbol{Y}$. By construction this step is equivalent to computing optimal warping paths for determining the DTW distance between $\boldsymbol{x}_i$ and $\boldsymbol{y}$. Line 2.2 updates matrix $\boldsymbol{Y}$ and line 2.3 projects the updated matrix $\boldsymbol{Y}$ to the set $\mathcal{Z}$. The last step is equivalent to constructing a time series from a matrix.

Previous approaches differ in the form of update rule υ in line 2.2 and the projection π in line 2.3. Algorithmically, steps 2.2 and 2.3 usually form a single step in the sense that the composition $\psi = \pi \circ \upsilon$ can not as clearly decomposed in two separate processing steps as described in Algorithm 2. The choice of υ and π is critical for convergence analysis. Problems arise when the map υ does not select a generalized gradient and the projection π does not map a matrix from $\mathcal{X}$ to a closest matrix from $\mathcal{Y}$. In these cases, it may be unclear how to define necessary conditions of optimality for the function f. As a consequence, even if steps 2.2 and 2.3 minimize f, we do not know whether Algorithm 2 converges to a local minimum of f. The same problems arise when studying the asymptotic properties of the mean as a minimizer of f.

The situation is different for the function F defined in eq. (10). When minimizing F, the set $\mathcal{Z}$ coincides with $\mathcal{X}$. Since the function F is piecewise smooth, the map υ in line 2.2 corresponds to an update step of the generalized gradient method. The projection π in line 2.3 is the identity. Under the conditions of [5], Theorem 4.1 and Theorem 5.1 the procedure described in Algorithm 2 is consistent.

Dataset	#(Train)	#(Test)	Length	ρ
Coffee	28	28	286	0.098
ECG200	100	100	96	1.042
ECGFiveDays	23	861	136	0.169
Gun Point	50	150	150	0.333
ItalyPowerDemand	67	1,029	24	2.792
Lightning 2	60	61	637	0.094
MoteStrain	20	1,252	84	0.238
SonyAIBORobotSurface	20	601	70	0.286
SonyAIBORobotSurfaceII	27	953	65	0.415
TwoLeadECG	23	1,139	82	0.280
Wafer	1,000	6,174	152	6.579
Yoga	300	3,000	426	0.704

Table 2: Characteristic features of data sets for two-class classification problems. The last column shows the ratio $\rho = \text{length}/\#(\text{train})$.

5. Experiments

The goal of this section is to assess the performance and behavior of elastic linear classifiers.We present and discuss results from two experimental studies. The first study explores the effects of the elasticity parameter on the error rate and the second study compares the performance of different elastic linear classifiers. We considered two-class problems of the UCR time series datasets [10]. Table 2 summarizes characteristic features of the datasets.

5.1. Exploring the Effects of Elasticity

The first experimental study explores the effects of elasticity on the error rate by controlling the number of columns of the weight matrix of an elastic perceptron.

5.1.1. Experimental Setup.

The elastic perceptron algorithm was applied to the Gun_Point, ECG200, and ECG-FiveDays dataset using the following setting: The dimension of the matrix space $\mathcal{X}$ was set to $n \times m$, where n is the length of the longest time series in the training set of the respective dataset. Bias and weight matrix were initialized by drawing random numbers from the uniform distribution on the interval $[-0.01, +0.01]$. The elasticity m was controlled via the ratio $w = m/n$. For every $w \in \mathcal{S}_w$ the learning rate $\eta \in \mathcal{S}_\eta$ with the lowest error on the training set was selected, where the sets are of the form

$$\mathcal{S}_w = \{0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0, 2.0, 3.0\}$$
$$\mathcal{S}_\eta = \{1.0, 0.7, 0.3, 0.1, 0.03, 0.01, 0.003, 0.001\}\,.$$

Note that the value $w = 0$ refers to $m = 1$. Thus the weight matrix collapses to a column vector and the elastic perceptron becomes the standard perceptron. To assess the generalization performance, the learned classifier was applied to the test set. The whole experiment was repeated 30 times for every value w.

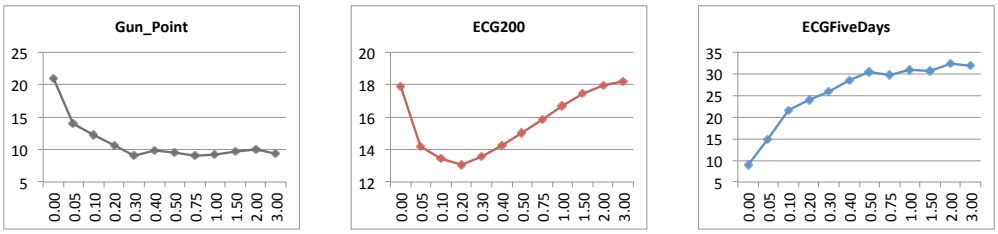

Figure 4: Mean error rates of elastic perceptron on Gun_Point, ECG200, and ECGFiveDays. Vertical axes show the mean error rates in % averaged over 30 trials. Horizontal axes show the ratio $w = m/n$, where m is the elasticity, that is the number of columns of the weight matrix and n is the length of the longest time series of the respective dataset. Ratio $w = 0$ means $m = 1$ and corresponds to the standard perceptron algorithm.

5.1.2. Results and Discussion

Figure 4 shows the mean error rates of the elastic perceptron as a function of $w = m/n$. The error rates on the respective training sets were always zero.

One characteristic feature of the UCR datasets listed in Table 2 is that the number of training examples is low compared to the dimension of the time series. This explains the low training error rates and the substantially higher test error rates.

The three plots show typical curves also observed when applying the elastic perceptron to the other datasets listed in Table 2. The most important observation to be made is that the parameter w is problem-dependent and need to be selected carefully. If the training set is small and dimensionality is high, a proper choice of w becomes challenging. The second observation is that in some cases, the standard perceptron algorithm ($w = 0$) may perform best as in ECGFiveDays. Increasing w results in a classifier with larger flexibiltiy. Intuitively this means that an elastic perceptron can implement more decision boundaries the larger w is. If w becomes too large, the classifier becomes more prone to overfitting as indicated by the results on ECG200 and ECGFiveDays. We hypothesize that elasticity controls the capacity of an elastic linear classifier.

5.2. Comparative Study

This comparative study assesses the performance of elastic linear classifiers.

5.2.1. Experimental Setup.

In this study, we used all datasets listed in Table 2. The four elastic linear classifiers of Section 3.4 were compared against different variants of the nearest neighbor (NN) classifier with DTW distance. The variants of the NN classifiers differ in the choice of prototypes. The first variant uses all training examples as prototypes (NN+ALL). The second and third variant learned one prototype per class from the training set

using k-means (NN+KME) as second variant and agglomerative hierarchical clustering (NN+AHC) as third variant [18].

The settings of the elastic linear classifiers were as follows: The dimension of the matrix space $\mathcal{X}$ was set to $n \times m$, where n is the length of the longest time series in the training set and $m = \lceil n/10 \rceil$ is the elasticity. The elasticity m was set to 10% of the length n for the following reasons: First, m should be small to avoid overfitting due to high dimensionality of the data and small size of the training set. Second, m should be larger than one, because otherwise an elastic linear classifier reduces to a standard linear classifier.

Bias and weight matrix were initialized by drawing random numbers from the uniform distribution on the interval $[-0.01, +0.01]$. Parameters were selected by k-fold cross validation on the training set of size N. We set $k = 10$ if $N > 30$ and $k = N$ otherwise. The following parameters were selected: learning rate η for all elastic linear classifiers, margin ξ for elastic margin perceptron, and regularization parameter λ for elastic linear SVM. The parameters were selected from the following values

$$\eta \in \left\{2^{-10}, 2^{-9}, \ldots, 2^{0}\right\}, \quad \xi \in \left\{10^{-7}, 10^{-6}, \ldots, 10^{1}\right\}, \quad \lambda \in \left\{2^{-10}, 2^{-9}, \ldots, 2^{-1}\right\}.$$

The final model was obtained by training the elastic linear classifiers on the whole training set using the optimal parameter(s). We assessed the generalization performance by applying the learned model to the test data. Since the performance of elastic linear classifiers depends on the random initialization of the bias and weight matrix, we repeated the last two steps 100 times, using the same selected parameters in each trial.

5.2.2. Results and Discussion.

Table 3 summarizes the error rates of all elastic linear (EL) classifiers and nearest neighbor (NN) classifiers.

Comparison of EL classifiers and NN methods is motivated by the following reasons: First, NN classifiers belong to the state-of-the-art and are considered to be *exceptionally difficult to beat* [1, 12, 29]. Second, in Euclidean spaces linear classifiers and nearest neighbors are two simple but complementary approaches. Linear classifiers are computationally efficient, make strong assumptions about the data and therefore may yield stable but possibly inaccurate predictions. In contrast, nearest neighbor methods make very mild assumption about the data and therefore often yield accurate but possibly unstable predictions [8].

The first key observation suggests that overall generalization performance of EL classifiers is comparable to the state-of-the-art NN classifier. This observation is supported by the same same number of green shaded rows (EL is better) and red shaded rows (NN is better) in Table 3. As reported by [12], ensemble classifiers of different elastic distance measures are assumed to be first approach that significantly outperformed the NN+ALL classifier on the UCR time series dataset. This result is not surprising, because in machine learning it is well known for a long time that ensemble classifiers often perform better than their base classifiers for reasons explained in [4].

Dataset	NN + DTW			Elastic linear classifiers			
	ALL	AHC	KME	ePERC	eLOGR	eMARG	eLSVM
Coffee	17.9	25.0	25.0	$4.6^{\pm 3.0}$	$4.5^{\pm 2.9}$	$4.7^{\pm 3.3}$	$\mathbf{3.1}^{\pm 2.9}$
ECG200	23.0	28.0	28.0	$13.6^{\pm 1.8}$	$\mathbf{11.8}^{\pm 1.6}$	$13.6^{\pm 1.9}$	$13.1^{\pm 1.7}$
ECGFiveDays	23.2	33.0	33.0	$15.3^{\pm 3.7}$	$15.7^{\pm 3.3}$	$15.3^{\pm 3.4}$	$\mathbf{11.1}^{\pm 3.0}$
ItalyPowDem.	5.0	21.5	21.5	$3.8^{\pm 1.2}$	$\mathbf{3.0}^{\pm 0.3}$	$3.5^{\pm 0.8}$	$\mathbf{3.0}^{\pm 0.3}$
Wafer	2.0	69.5	69.5	$1.3^{\pm 0.3}$	$1.2^{\pm 0.2}$	$\mathbf{1.0}^{\pm 0.2}$	$\mathbf{1.0}^{\pm 0.2}$
Gun Point	9.3	32.7	32.7	$9.7^{\pm 3.6}$	$9.2^{\pm 2.5}$	$10.0^{\pm 3.4}$	$\mathbf{9.0}^{\pm 2.8}$
SonyAIBO II	27.5	21.6	21.6	$27.0^{\pm 3.8}$	$\mathbf{20.2}^{\pm 1.4}$	$26.6^{\pm 3.3}$	$22.7^{\pm 2.1}$
Lighting 2	**13.1**	36.1	36.1	$44.2^{\pm 4.2}$	$44.1^{\pm 4.4}$	$44.6^{\pm 4.4}$	$47.6^{\pm 2.9}$
MoteStrain	16.5	**13.3**	**13.3**	$17.2^{\pm 2.6}$	$16.0^{\pm 2.3}$	$17.6^{\pm 2.6}$	$15.8^{\pm 2.3}$
SonyAIBO	**16.9**	18.8	18.8	$19.3^{\pm 5.4}$	$18.6^{\pm 5.0}$	$19.5^{\pm 6.6}$	$17.8^{\pm 4.2}$
TwoLeadECG	**9.6**	16.2	16.2	$22.7^{\pm 5.3}$	$21.8^{\pm 4.5}$	$21.7^{\pm 5.4}$	$21.8^{\pm 5.3}$
Yoga	**16.4**	45.8	45.8	$20.9^{\pm 1.2}$	$21.5^{\pm 1.0}$	$21.1^{\pm 1.1}$	$20.8^{\pm 1.1}$

Table 3: Mean error rates and standard deviation of elastic linear classifiers averaged over 100 trials and error rates of nearest-neighbor classifiers using the DTW distance (NN+DTW). ALL: NN+DTW with all training examples as prototypes; AHC: NN+DTW with one prototype per class obtained from agglomerative hierarchical clustering with Ward linkage; KME: NN+DTW with one prototype per class obtained from k-means clustering; ePERC: elastic perceptron; eLOGR: elastic logistic regression; eMARG = elastic margin perceptron; eLSVM: elastic linear SVM. Best (avg.) results are highlighted. Green rows: avg. results of all elastic linear classifiers are better than the results of all NN classifiers. Yellow rows: results of elastic linear classifiers and NN classifiers are comparable. Red rows: avg. results of all elastic linear classifiers are worse than the best result of an NN classifier.

Since any base classifier can contribute to an ensemble classifier, it is feasible to restrict comparison to base classifiers such as the state-of-the-art NN+ALL classifier.

The second key observation indicates that EL classifiers are clearly superior to NN classifiers with one prototype per class, denoted by NN_1 henceforth. Evidence for this finding is provided by two results: first, AHC and KME performed best among several prototype selection methods for NN classification [18]; and second, error rates of EL classifiers are significantly better than those of NN+AHC and NN+KME for eight, comparable for two, and significantly worse for two datasets.

The third key observation is that EL classifiers clearly better compromise between solution quality and computation time than NN classifiers. Findings reported by [27] indicate that more prototypes may improve generalization performance of NN classifiers. At the same time, more prototypes increase computation time, though the differences will decrease for larger number of prototypes by applying certain acceleration techniques. At the extreme ends of the scale, we have NN+ALL and NN_1 classifiers. With respect to solution quality, the first key observation states that EL classifiers are comparable to the slowest NN classifiers using the whole training set as prototypes and clearly superior to the fastest NN classifiers using one prototype per class. To compare computational efficiency, we first consider the case without applying any acceleration techniques. We measure computational efficiency by the number

of proximity calculations required to classify a single time series. This comparison is justified, because the complexity of computing a DTW distance and an elastic inner product are identical. Then EL classifiers are p-times faster than NN classifiers, where p is the number of prototypes. Thus the fastest NN classifiers effectively have the same computational effort as EL classifiers for arbitrary multi-class problems, but they are not competitive to EL classifiers according to the second key observation. Next, we discuss computational efficiency of both types of classifiers, when one applies acceleration techniques. For NN classifiers, two common techniques to decrease computation time are global constraints such as the Sakoe-Chiba band [23] and diminishing the number of DTW distance calculations by applying lower bounding technique [21, 22]. Both techniques can equally well be applied to EL classifiers, where lower-bounding techniques need to be converted to upper-bounding techniques. Furthermore, EL classifiers can additionally control the computational effort by the number m of columns of the matrix space. Here m was set to 10% of the length n of the shortest time series of the training set. The better performance of EL classifiers in comparison to NN_1 classifiers is notable, because the decision boundaries that can be implemented by their counterparts in the Euclidean space are both the set of all hyperplanes. We assume that EL classifiers outperform NN_1 classifiers, because learning prototypes by clustering minimizes a cluster criterion unrelated to the risk functional of a classification problem. Therefore the resulting prototypes may fail to discriminate the data for some problems.

The fourth key observation is that the strong assumption of elastic-linearly separable problems is appropriate for some problems in the time series classification. Error rates of elastic linear classifiers for Coffee, ItalyPowerDemand, and Wafer are below 5%. For these problems, the strong assumption made by EL classifiers is appropriate. For all other datasets, the high error rates of EL classifiers could be caused by two factors: first, the assumption that the data is elastic-linearly separable is inappropriate; and second, the number of training examples given the length of the time series is too low for learning (see ratio ρ in Table 2). Here further experiments are required.

The fifth observation is that the different EL classifiers perform comparable with advantages for eLOGR and eLSVM. These findings correspond to similar findings for logistic regression and linear SVM in vector spaces.

To complete the comparison, we contrast the time complexities of all classifiers required for learning. NN+ALL requires no time for learning. The NN+AHC classifier learns a protoype for each class using agglomerative hierarchical clustering. Determining pairwise DTW distances is of complexity $O(n^2N(N-1)/2)$, where n is the length of the time series and N is the number of training examples. Given a pairwise distance matrix, the complexity of agglomerative clustering is $O(N^3)$ in the general case. Efficient variants of special agglomerative methods have a complexity of $O(N^2)$. Thus, the complexity of NN+AHC is $O(n^2N^2)$ in the best and $O(n^2N^2 + N^3)$ in the general case. The NN+KME learns a protoype for each class using k-means under elastic transformations. Its time complexity is $O(2n^2Nt)$, where t is the number of iterations required until termination. The time complexity for learning an EL classifier is $O(nmNt)$, where m is the number of columns of the weight matrix. This shows that the time complexity for learning an EL classifier is the same as learning

two prototypes by KME. However, in this setting, learning an EL classifier is about factor 20 faster than KME, under the assumption that the number of iterations t is the same for both methods. If the number N of training examples is large, NN+AHC becomes prohibitively slow. In contrast, the learning procedures of NN+KME and EL classifiers can be terminated after some pre-specified maximum number of iterations. In doing so, we trade solution quality against feasible computation time.

To summarize, the results show that elastic linear classifiers are simple and efficient methods. They rely on the strong assumption that an elastic-linear decision boundary is appropriate. Therefore, elastic linear classifiers may yield inaccurate predictions when the assumptions are biased towards oversimplification and/or when the number of training examples is too low compared to the length of the time series. These findings are in line with those of linear classifiers in Euclidean space.

6. Conclusion

This paper introduces generalized gradient methods for learning on time series under elastic transformations. This approach combines (a) the novel concept of elastic functions that links elastic proximities on time series to piecewise smooth functions with (b) generalized gradient methods for non-smooth optimization. Using the proposed scheme, we (1) showed how a broad class of gradient-based learning can be applied to time series under elastic transformations, (2) derived general convergence statements that justify the generalizations, and (3) placed existing adaptive methods into proper context. Exemplarily, elastic logistic regression, elastic (margin) perceptron learning, and elastic linear SVM have been tested on two-class problems and compared to nearest neighbor classifiers using the DTW distance. Despite the simplicity in terms of the decision boundary and the computational efficiency, elastic linear classifiers perform convincing. There is still room for improvement by controlling elasticity and by applying different forms of regularization. The results indicate that adaptive methods based on elastic functions may complement the state-of-the-art in statistical pattern recognition on time series, in particular when powerful non-linear gradient-based methods such as deep learning are extended to time series under elastic transformations.

References

[1] G.E. Batista, X. Wang, and E.J. Keogh. A Complexity-Invariant Distance Measure for Time Series. *SIAM International Conference on Data Mining*, 11:699–710, 2011.

[2] L. Bottou. Stochastic learning. *Advanced Lectures on Machine Learning*, Springer, 200.

[3] F.H. Clarke. Generalized gradients and applications. *Transactions of the American Mathematical Society*, 205: 247–262, 1975.

[4] T.G. Dietterich. Ensemble methods in machine learning. *Proceedings of the First International Workshop on Multiple Classifier Systems*, 2000.

[5] Y. Ermoliev and V. Norkin. Stochastic generalized gradient method for nonconvex nonsmooth stochastic optimization. *Cybernetics and Systems Analysis*, 34(2):196–215, 1998.

[6] T. Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[7] P. Geurts. Pattern extraction for time series classification. *Principles of Data Mining and Knowledge Discovery*, pp. 115–127, 2001.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2001.

[9] V. Hautamaki, P. Nykanen, and P. Franti. Time-series clustering by approximate prototypes. *International Conference on Pattern Recognition*, 2008.

[10] E. Keogh, Q. Zhu, B. Hu, Y. Hao., X. Xi, L. Wei, and C. A. Ratanamahatana. The UCR Time Series Classification/Clustering Homepage: `www.cs.ucr.edu/~eamonn/time_series_data/`, 2011.

[11] J.B. Kruskal and M. Liberman. The symmetric time-warping problem: From continuous to discrete *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*, p. 125–161, 1983

[12] J. Lines and A. Bagnall. Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, 2014.

[13] A. Nedic and D.P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12(1):109–138, 2001.

[14] V. Niennattrakul and C.A. Ratanamahatana. Inaccuracies of shape averaging method using dynamic time warping for time series data. *International Conference on Computational Science*, pp. 513–520, 2007.

[15] V. Niennattrakul and C.A. Ratanamahatana. On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping *International Conference on Multimedia and Ubiquitous Engineering*, pp. 733–738, 2007.

[16] V. Norkin. Stochastic generalized-differentiable functions in the problem of nonconvex nonsmooth stochastic optimization. *Cybernetics and Systems Analysis*, 22(6):804–809, 1986.

[17] F. Petitjean, A. Ketterlin, and P. Gancarski. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, 44(3): 678–693, 2011.

[18] F. Petitjean, G. Forestier, G.I. Webb, A.E. Nicholson, Y. Chen, and E. Keogh. Dynamic Time Warping Averaging of Time Series allows Faster and more Accurate Classification. *International Conference on Data Mining*, 2014.

[19] L.R. Rabiner and J.G. Wilpon. Considerations in applying clustering techniques to speaker?independent word recognition. *The Journal of the Acoustical Society of America*, 66(3): 663–673, 1979.

[20] L.R. Rabiner and J.G. Wilpon. A simplified, robust training procedure for speaker trained, isolated word recognition systems. *The Journal of the Acoustical Society of America*, 68(5):1271–1276.

[21] C. A. Ratanamahatana and E. J. Keogh. Making time- series classification more accurate using learned constraints. *SIAM International Conference on Data Mining*, 2004.

[22] C. A. Ratanamahatana and E. J. Keogh. Three myths about dynamic time warping data mining. *SIAM International Conference on Data Mining*, pp. 506–510, 2005.

[23] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[24] N.Z. Shor. *Minimization Methods for Nondifferentiable Functions*. Springer, 1985.

[25] P. Somervuo and T. Kohonen. Self-organizing maps and learning vector quantization for feature sequences. *Neural Processing Letters*, 10(2): 151–159, 1999.

[26] V. Vapnik, E. Levin, and Y. Le Cun. Measuring the VC-dimension of a learning machine. *Neural Computation*, 6(5): 851–876, 1994.

[27] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2): 275–309, 2013.

[28] J.P. Wilpon and L.R. Rabiner. A modified K-means clustering algorithm for use in isolated work recognition. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 33(3): 587–594, 1985.

[29] X. Xi, E. Keogh, C. Shelton, L. Wei, and C.A. Ratanamahatana. Fast time series classification using numerosity reduction. *International Conference on Machine Learning*, pp. 1033–1040, 2006.

A. Proof of Convergence Results for Elastic Linear Classifiers

Since affine functions are convex and the maximum of convex functions is also convex, the elastic inner product is convex. In addition, the composition of convex functions is convex. Therefore the loss functions of elastic linear classifiers are convex. Then the first convergence results is shown in [24].

To show the two other convergence statements, we assume that $|\mathcal{D}| = N$. For each training example $(\boldsymbol{x}_i, y_i) \in \mathcal{D}$ the loss

$$\ell_i(\boldsymbol{\theta}) = \ell_i\left(y_i, b + \sigma(\boldsymbol{x}_i, \boldsymbol{W})\right)$$

is real-valued and convex, where $\boldsymbol{\theta} = (\boldsymbol{W}, b)$. Then there is a positive scalar C_i that bounds the subdifferential of ℓ_i at $\boldsymbol{\theta}$ for all $i \in [N]$. Suppose that

$$C = \max_{i=1,\ldots,N} C_i.$$

Then from [13], Prop. 2.2. follows that the incremental generalized gradient method converges to a local minimum.

To show the Elastic Margin Perceptron Convergence Theorem, we assume that

$$E_N[\boldsymbol{\theta}] = \sum_{i=1}^{N} \ell_i(\boldsymbol{\theta})$$

is the error without averaging operation, that is $E_N = N \cdot R_N$. By assumption, the training set $\mathcal{D}$ is elastic-linearly separable. Then the minimum value E_* of E_N is zero. From [13], Prop. 2.1. follows

$$\lim_{t\to\infty} E_N(\boldsymbol{\theta}^t) \leq E_* + \frac{\eta \cdot C^2}{2} = \frac{\eta \cdot C^2}{2},$$

where η is the learning rate. Choosing $\eta \leq \xi / C^2$ gives

$$\lim_{t\to\infty} E_N(\boldsymbol{\theta}^t) \leq \frac{\xi}{2}.$$

Since $\xi > 0$, this implies that there is a t_0 such that $\ell_t(\boldsymbol{\theta}^t) < \xi$ for all $t \geq t_0$. Here, ℓ_t refers to example $(\boldsymbol{x}_t, y_t) \in \mathcal{D}$ presented at iteration t. From this follows that all training examples are classified correctly after a finite number of update steps, provided that $\lambda \leq \xi$. ■